Wuhan University
Journal of Natural Sciences

2018, Vol.23 No.3, 237-243

Article ID 1007-1202(2018)03-0237-07
DOI https://doi.org/10.1007/s11859-018-1316-z

Sentiment Analysis of Code-Mixed Bambara-French Social Media Text Using Deep Learning Techniques

☐ **Arouna KONATE[1], DU Ruiying[1,2†]**

1. State Key Laboratory of Software Engineering/School of Computer, Wuhan University, Wuhan 430072, Hubei, China;

2. Collaborative Innovation Center of Geospatial Technology, Wuhan 430079, Hubei, China

Abstract: The global growth of the Internet and the rapid expansion of social networks such as Facebook make multilingual sentiment analysis of social media content very necessary. This paper performs the first sentiment analysis on code-mixed Bambara-French Facebook comments. We develop four Long Short-term Memory (LSTM)-based models and two Convolutional Neural Network (CNN)-based models, and use these six models, Naïve Bayes, and Support Vector Machines (SVM) to conduct experiments on a constituted dataset. Social media text written in Bambara is scarce. To mitigate this weakness, this paper uses dictionaries of character and word indexes to produce character and word embedding in place of pre-trained word vectors. We investigate the effect of comment length on the models and perform a comparison among them. The best performing model is a one-layer CNN deep learning model with an accuracy of 83.23 %.

Key words: sentiment analysis; code-mixed Bambara-French Facebook comments; deep learning; Long Short-Term Memory(LSTM); Convolutional Neural Network (CNN)

CLC number: TP 391.1

Received date: 2017-11-16

Foundation item: Supported by the National Natural Science Foundation of China (61272451, 61572380, 61772383 and 61702379), and the Major State Basic Research Development Program of China (2014CB340600)

Biography: Arouna KONATE, male, Ph. D. candidate, research direction: information security, deep learning, natural language processing. E-mail: arounakonate@yahoo.fr

† To whom correspondence should be addressed. E-mail: duraying@126.com

0 Introduction

Communication is generating a new type of content due to the expanding access of the Internet and the growing adoption of social media, such as Facebook and Twitter, all over the world. Some of the contents on these social media sites are informal and are often written in two or more languages with users switching between languages in a single sentence. In many West African countries (including Mali, Burkina-Faso, Cote-d'Ivoire, Senegal and Guinea), Bambara is a national language alongside French which plays the role of official language. As a result, social media contents in these countries are often language mixing that particularly contain Bambara and French.

Language mixing is referred to as code-mixing or code switching[1,2]. Unique elements such as intra-sentential code mixing (within a sentence), inter-sentential (across sentences) code-mixing, creative spellings, lexical borrowings and phonetic typing are the characteristics of social media text.

The nature of the global Internet with all sorts of activities taking place (criminal, hatred and terrorist activities) and the use of almost every language in the world on social media make it necessary to analyze online content in a multilingual context. One such type of analysis is sentiment analysis, which aims to extract and analyze opinion and subjectivity knowledge presented in online text[3]. Sentiment analysis is performed using either machine learning or lexical-based methods. The success of machine learning approaches depends on the quality and quantity of labelled corpora. Most researches

on sentiment analysis have focused on languages such as English, Spanish and Chinese, for which large labelled corpora (monolingual or multilingual) are available.

For languages such as Bambara of which written text on the Internet is scarce, corpora (monolingual or multilingual) virtually do not exist. The first step is therefore the acquisition of a labelled corpus suitable for the task. This article starts with the preparation of a labelled three-class sentiment dataset (positive, neutral and negative) from Facebook comments. The next step is using techniques to extract features for further processing. Common features extracted from corpora are bag-of-words representations, pre-trained word and character embedding. Word embedding is generally representations obtained by pre-training them on large corpora. Compared to traditional machine learning techniques using bag-of-words features, deep learning techniques exploiting pre-trained word embedding achieve very encouraging performance on classification tasks. With languages such as Bambara, such pre-trained word representations cannot be of any good use, for the simple reason that they are obtained from very small corpora. Alternative character and word representations are necessary in such circumstances.

Currently, apart from Bambara dictionary corpus building research [4], there is no research work either on Bambara or Bambara-French code-mixed social media text sentiment analysis, and there is no sentiment labelled corpus available for Bambara or Bambara-French social media text.

This paper performs for the first time sentiment analysis on code-mixed Bambara-French social media text. We use character and word representations obtained from fixed indexes instead of pre-trained word embedding to produce sentence and comment representations.

The aim of the work in this article is threefold:

1) We develop six deep learning models for the first sentiment analysis of code-mixed Bambara-French Facebook comments. We also compare the effectiveness of character and word vectors produced from fixed indexes to form sentence-level representations for this task.

2) We investigate the effect of comment length on both the deep learning methods and well-known traditional machine learning algorithms.

3) We compare the deep learning techniques and the classical machine learning algorithms Naïve Bayes and SVM, and ultimately propose the best learning model for this task.

The rest of the paper is organized as follows. Section 1 is about related work. Section 2 contains descriptions of the proposed deep learning models and the character and word representations. Section 3 reports the data collection, preprocessing process, the experiments and results analysis. Section 4 concludes this paper.

1 Related Work

Sentiment analysis is performed as a classification task with three levels of classification which are document-level sentiment analysis [5], sentence-level sentiment-analysis [5,6] and target or aspect-level sentiment analysis [7]. Document-level sentiment analysis determines the sentiment (positive, neutral, negative) expressed in a document by considering the whole document as a basic information unit. For sentence-level sentiment analysis, the subjectivity of a sentence is analyzed. Aspect-level sentiment analysis classifies sentiment of a specific aspect in a sentence or document. The sentiment analysis task performed in this paper focuses exclusively on sentence-level sentiment analysis, and we will use machine-learning techniques instead of lexical-based methods.

Machine learning techniques used for sentiment analysis are traditional machine learning algorithms, such as Naive Bayes and Support Vector Machines (SVM)[8, 9], and deep learning techniques. They are mostly concerned with monolingual corpora. Traditional machine learning algorithms are generally used in combination with handcrafted features requiring intense manual labor and domain knowledge [10,11]. A Long Short-term Memory (LSTM) is a Recurrent Neural Network (RNN) that was introduced in Ref.[12] and Convolutional Neural Network (CNN) was first introduced in Ref. [13] for the task of document recognition. An LSTM with trainable look-up table proposed by Wang *et al*[14] achieved state-of-the-art performance on Twitter sentiment. CNN was used for sentiment analysis on Twitter data in Refs. [15], [16] and [17]. A study of deep learning on Thai Twitter sentiment analysis has compared LSTM and Dynamic Convolution Neural Network (DCNN) using pre-trained word vectors against Naïve Bayes, SVM and Maximum Entropy[18], and a Bidirectional Long Short-term Memory (BLSTM) model is used for sentiment analysis of Chinese social media text[3].

Code switching or code mixing is a linguistic phenomenon that has gained much attention over the decades. However, it was only in the 1980s that the first computational work was performed by Joshi[19]. Most

researches on the use of deep learning on code-mixed social media text for Natural Language Processing (NLP) purposes have focused on language identification. Chang *et al*[19] have combined RNN with pre-trained word embedding and their system outperformed the best SVM-based systems on the EMNLP 2014 code-switching language identification workshop. Samih *et al*[20] combined character with word pre-trained vectors and used LSTM instead of RNN. Their submitted system for the EMNLP 2016 workshop on computational approaches to code-switching came first for Arabic (MSA-Egyptian) and ranked second for English-Spanish. Santos *et al*[17] and Collobert *et al*[21] argued that character embedding can capture word morphology and reduce out-of-vocabulary making them effective for many NLP tasks. Kim *et al* [22] showed that character embedding is suitable for handling languages with rich morphology and large character sets. Similar to Ref. [20], this paper uses character and word representations to form code-mixed text sentence representations. However, this paper leverages fixed indexes to obtain character and word embedding in place of pre-trained word vectors.

2 Models

In this paper, we developed the following six deep learning models: one-layer LSTM, two-layer LSTM one-layer BLSTM, two-layer BLSTM, one-layer CNN and CNN-LSTM. Specifically, each model has an embedding layer, one or more LSTM/BLSTM/CNN layers, one or more dropout layers and a softmax layer. For comparison, the same embedding layer was used for all the models, and dropout[23] layers were applied to optimize the models.

2.1 LSTM/BLSTM-Based Models

The architecture of the LSTM/BLSTM-based models is given in Fig. 1. Specifically, the embedding layer takes the words w_t ($t \in [1, N]$) of a sentence or comment and an embedding matrix $\boldsymbol{W}_e$ as input, and embeds them to vectors $\boldsymbol{x}_t = \boldsymbol{W}_e w_t$. In the LSTM-based models, we add one or two LSTM layers on top of the embedding layer.

In the one-layer LSTM model, we apply dropout to optimize the vectors $\boldsymbol{x}_t$ output by the embedding layer. Afterwards, an LSTM layer takes these $\boldsymbol{x}_t$ vectors and produces hidden states $\boldsymbol{h}_t$ by computing the following calculations:

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{ci}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i\right) \tag{1}$$

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{cf}\boldsymbol{c}_{t-1} + \boldsymbol{b}_f\right) \tag{2}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t\boldsymbol{c}_{t-1} + \boldsymbol{i}\tanh\left(\boldsymbol{W}_{xc}\boldsymbol{x}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right) \tag{3}$$

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{co}\boldsymbol{c}_t + \boldsymbol{b}_o\right) \tag{4}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t\tanh\left(\boldsymbol{c}_t\right) \tag{5}$$

where $\boldsymbol{f}$, $\boldsymbol{i}$, $\boldsymbol{o}$, $\boldsymbol{c}$ and σ denote the forget gate, the input gate, the output gate, the cell activation vectors, and the logistic sigmoid function, respectively. We apply a second dropout layer to the output of the LSTM layer and then add a dense layer on top of the dropout layer to produce the sentence representation

$$\boldsymbol{s} = \mathrm{relu}\left(\boldsymbol{h}_t\right) \tag{6}$$

This sentence representation is fed into a softmax layer to classify it as conveying positive, neutral or negative sentiment.

$$\boldsymbol{p} = \mathrm{softmax}\left(\boldsymbol{s}\right) \tag{7}$$

In the two-layer LSTM model, an LSTM layer is added on top of the first LSTM layer.

A BLSTM is a bidirectional LSTM that uses a forward LSTM and a backward LSTM to capture strong distance dependencies by leveraging past and future information. The one-layer and two-layer BLSTM models follow the same architecture as the LSTM-based models by replacing the LSTM layers with BLSTM layers.

2.2 Convolutional Neural Network (CNN)-Based Models

The architecture of the CNN-based models is given in Fig. 2. Specifically, in the one-layer CNN model a

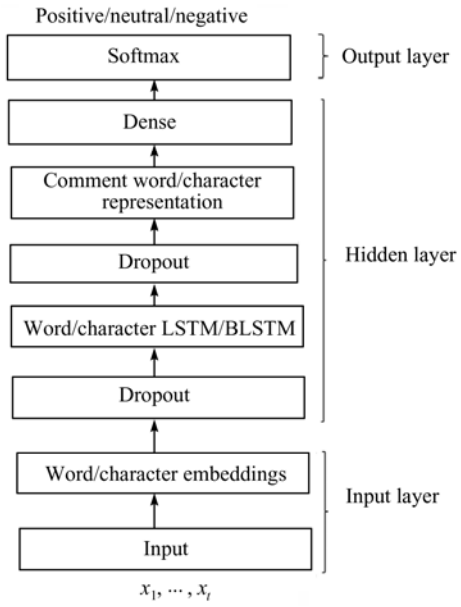

Fig.1 Architecture of LSTM/BLSTM-based models

dropout layer is applied to the vectors $\boldsymbol{x}_t = \boldsymbol{W}_e w_t$ output by the embedding layer. A convolutional layer takes the output of the dropout layer and performs convolutional operations as follows:

$$\boldsymbol{c}_{ij}^{l} = \boldsymbol{f}\left(\sum_{z\in\mathbf{Z}} \boldsymbol{c}_{ij}^{l-i} * \boldsymbol{W}_{ij}^{l} + \boldsymbol{b}_{j}^{l}\right) \tag{8}$$

where $z \in \mathbf{Z}$ represents the convolutional windows, $\boldsymbol{W}_{ij}$ is the kernel matrix, and $\boldsymbol{b}$ is the bias vector.

A max-pooling layer on top of the convolutional layer uses pooling windows $k \in K$ and a pooling value to compute its output and takes the output of the convolutional layer to produce its own output as follows:

$$\boldsymbol{c}_{ij}^{l} = \boldsymbol{f}(\max_{k\in K} \boldsymbol{c}_{ij}^{l-1} + \boldsymbol{b}_{j}^{l}) \tag{9}$$

We add a fully connected layer on top of the max-pooling layer before a softmax function that decides whether the sentiment of the sentence is positive, neutral or negative. The convolutional and pooling layers capture the different sentiment features presented in a sentence, and their output are obtained as low dimensional vectors through the fully connected layer. At the convolutional layer, features are extracted as feature maps, convolved with learnable kernels, and pass through an activation function that produces the convolutional layer output feature maps.

In the CNN-LSTM architecture, an LSTM layer is used after the pooling layer without adding a fully connected layer before the softmax layer in this model.

2.3 Character and Word Embedding

In contrast to the traditional machine learning models, all of our proposed deep learning models embed the inputs through a common layer using either character or word embedding.

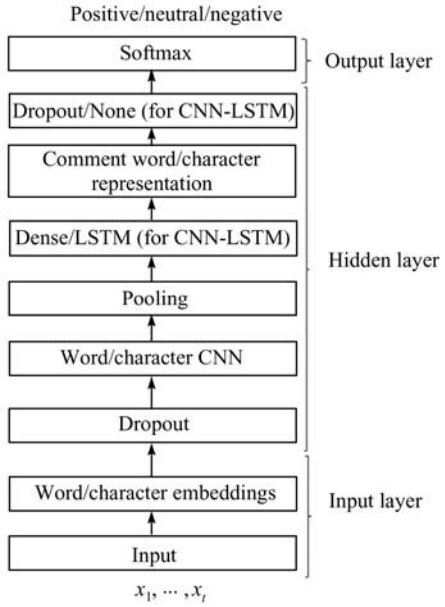

Fig.2 Architecture of CNN/CNN-LSTM-based models

Compared with languages such as English, no pre-trained word vectors are available for Bambara. As a result, we first extract the unique characters forming the character set of the code-mixed corpus, and then build a dictionary associating fixed indexes (fixed positive integers), by which dense vectors for the characters can be produced.

Similar to the character embedding, we build a dictionary associating fixed indexes for the unique words in the code-mixed corpus to produce dense vectors for the words.

3 Experiments and Result Analysis

3.1 Data Collection and Preprocessing

We collected and preprocessed Facebook comments to obtain the social media labelled sentiment data. Specifically, with Facebook graph API Explorer, more than 74 000 code-mixed Bambara-French Facebook comments from Malian Facebook users were collected. We then randomly selected 20 000 comments. Subsequently, by removing all emoticons, digits, numbers and characters that are not in Bambara and French character sets and all comments in other languages, 17 062 comments in Bambara and French were eventually selected and manually labelled as positive, neutral or negative (shown in Table 1).

Table 1 Statistics of the dataset

Total	Positive	Negative	Neutral
17 062	7 707	3 232	6 123

3.2 Experiments

We divided the dataset into three parts: train (70% of the dataset), validation (20% of the dataset) and test (10% of the dataset) [24,25]. This splitting gives 11 943 comments as training set, 1 707 comments constitute the development set, and the test set contains 3 412 comments.

All the deep learning models proposed in this paper use the negative log likelihood of correct labels as training loss.

We measured all the results using the accuracy metric, which is defined as:

$$\text{accuracy} = \frac{\text{number of correct results}}{\text{number of total results}} \quad (10)$$

We trained the deep learning models for 2 epochs on the training and development sets with a batch size of 32 before testing them on the test set with a batch size of 64. The classical machine learning algorithms are tested after a 10-fold cross-validation[26]. The Theano [27] and Keras deep learning packages are used for the deep learning models while the scikit-learn[28] machine learning package is used for Naive Bayes and SVM.

Experiment 1 Investigation of the effect of comment length on the performance of the different models

In this experiment, we investigate the effect of comment length on the performance of the different models. Three comment lengths in terms of the number of words in a comment are considered, specifically lengths of 15 words, 20 words and 30 words. We report the best performance of each deep learning model using either the character or word embedding. The best performance of each classical model using either Term Frequency-Inverse Document Frequency (TF-IDF) word or character tri-gram (character sequences containing three characters each) representations is also reported in the results of the experiment.

Experiment 2 Comparison of the effect of the use of character and word embedding on the models

In this experiment, we test each proposed deep learning model using character and word embedding separately. TF-IDF word and character tri-gram representations are separately used as input for the classical models.

We conduct global comparison of the proposed deep learning models and the classical models

Specifically, the best accuracy of each proposed deep learning model on full-length comments using character and word embedding is reported. In addition, the best accuracy of each classical algorithm using TF-IDF and character tri-gram is reported.

3.3 Result Analysis

The results of Experiment 1 are shown in Table 2. The one-layer CNN deep learning model is the best model on comments of length shorter than 30 words. The two-layer BLSTM model is the winner on comments of 30 words in length. In the corpus, the longest comment has a length of 30 words corresponding to 192 characters. If all the comments have to obey the same length restriction as Tweets (140 characters), then the best model for sentiment analysis of code-mixed Bambara-French social media text would be one-layer CNN. Otherwise, for social media comments which are longer than those on Tweets, the two-layer BLSTM is the winner model.

Table 2 Accuracy of each of the models for each of the three comment lengths %

Model	Comment length		
	15 words[1]	20 words[2]	30 words[3]
One-layer LSTM	81.62	81.35	82.50
Two-layer LSTM	81.68	82.76	82.29
One-layer BLSTM	82.06	80.94	79.86
Two-layer BLSTM	81.70	82.85	**82.91**
One-layer CNN	**82.38**	**83.23**	82.12
CNN-LSTM	72.62	82.59	75.23
SVM	74.69	74.70	74.75
Naive Bayes	64.55	64.23	64.24

[1] or 117 characters; [2] or 154 characters; [3] or 192 characters

The results of Experiment 2 reported in Table 3 and Table 4 show that one-layer CNN and CNN-LSTM have achieved higher accuracy than the LSTM/BSLTM-based models. As for the classical models, Naïve Bayes performed slightly better on character tri-gram compared with word TF-IDF. However, similar to the deep learning models, SVM achieved a lower level of accuracy on character tri-gram compared with word TF-IDF. Moreover, we can find that all the models have achieved far lower level of accuracy by using character embedding than using word embedding, indicating that the character embedding defined in this paper is less appropriate than word embedding for deep learning.

Table 3 Accuracy of the deep learning models using character embedding and word embedding %

Model	Word embedding	Character embedding
One-layer LSTM	82.50	66.53
Two-layer LSTM	82.76	67.05
One-layer BLSTM	82.06	65.94
Two-layer BLSTM	82.91	65.38
One-layer CNN	**83.23**	**78.48**
CNN-LSTM	82.59	75.23

Table 4 Accuracy of classical algorithms using TF-IDF word and character tri-gram representations %

Model	TF-IDF words	Character tri-gram
SVM	**74.75**	**71.33**
Naive Bayes	64.25	64.55

Table 5 shows that, in terms of accuracy, the best model for sentiment analysis in this paper is one-layer CNN. Globally, all the deep learning models have outperformed the two classical machine-learning algorithms (Naive Bayes, SVM).

Table 5 Best overall accuracy of each model %

Model	Best overall accuracy
One-layer LSTM	82.50
Two-layer LSTM	82.76
One-layer BLSTM	82.06
Two-layer BLSTM	82.91
One-layer CNN	**83.23**
CNN-LSTM	82.59
SVM	74.75
Naive Bayes	64.55

The deep learning models outperformed the classical methods because deep learning models introduce more non-linearity in feature space. As a result, richer features are produced than the classical methods. The CNN-based models beat the other models because the convolutional layer in addition to the pooling layer capture the most important features for the present task. The LSTM-based models beat the classical methods, because they capture long dependencies between words and phrases. The two-layer BLSTM model outperforms one-layer and two-layer LSTM, because BLSTM learns previous and future information. In this way, it can capture even stronger dependencies between words and phrases than LSTM.

4 Conclusion

In this paper, we proposed six deep learning models, and applied them to the task of sentiment analysis of code-mixed Bambara-French Facebook comments. To supplement the absence of pre-trained word vectors for Bambara, we used character and word embedding produced from fixed indexes. The experimental results show that one-layer CNN is the best model. Moreover, the deep learning models outperformed the classical algorithms. In the future, we plan to gather more data and constitute larger corpora to train and test the use of pre-trained word embedding.

References

[1] Muysken P. *Code-Switching and Grammatical Theory*[M]. Cambridge: CUP, 1995.

[2] Gafaranga J, Torras M C. Interactional otherness: Towards a redefinition of code switching[J]. *International Journal of Bilingualism*, 2002 **6**(1): 1-22.

[3] Xiao Z, Liang P. *Chinese Sentiment Analysis Using Bidirectional LSTM with Word Embedding*[M]. Berlin Heidelberg: Springer-Verlag, 2016.

[4] Vydrin V. Bamana reference corpus (BRC)[J]. *Procedia-Soc Behav Sci*, 2013, **95**(4): 75-80.

[5] Balahur A, Steinberger R, Kabadjov M. Sentiment analysis in the news[J]. *Infrared Phys Technol*, 2014, **65**: 94-102.

[6] Long J, Yu M, Zhou M, *et al*. Target-dependent Twitter sentiment classification[C]// *Proceedings of the* 49*th Annual Meeting of the Association for Computational Linguistics*. Oregon: Association for Computational Linguistics, 2011: 151-160.

[7] Vaibhavi N P, Sheikh I R. Twitter as a corpus for sentiment analysis and opinion mining [J]. *International Journal of Advanced Research in Computer and Communication Engineering*, 2016, **5**(12):320-322.

[8] Go A, Bhayani R, Huang L. Twitter Sentiment Classification Using Distant Supervision[EB/OL]. [2018-03-14]. *https://www.researchgate.net/publication/*228523135_ *Twitter_ sentiment_classification_using_distant_supervision.*

[9] Pang B, Lee L, Vaithyanathan S. Thumbs up? Sentiment classification using machine learning techniques[C]// *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Philadelphia: Association for Computational Linguistics, 2002: 79-86.

[10] Bing L. *Sentiment Analysis and Opinion Mining*[M]. Williston: Morgan & Claypool Publishers, 2012.

[11] Mohammad S M, Kiritchenko S, Zhu X. NRC-Canada: Building the state-of-the-art in sentiment analysis of Tweets[C] //*Proceedings of the* 7*th International Workshop on Semantic Evaluation*. Atlanta: Association for Computational Linguistics, 2013: 321-327.

[12] Hochreiter S, Schmidhuber J. Long short-term memory[J]. *Neural Computation*, 1997, **9**(8): 1735-1780.

[13] Lecun Y, Bottou L, Bengio J, *et al*. Gradient-based learning

applied to document recognition[J]. *Proceedings of the IEEE*, 1998, **86**(11):2278-2324.

[14] Wang X, Liu Y, Sun C, *et al*. Predicting polarities of Tweets by composing word embeddings with Long Short-Term Memory[C]// *Proceedings of the* 53*rd Annual Meeting of the Association for Computational Linguistics and the* 7*th International Joint Conference on Natural Language Processing*. Beijing: Association for Computational Linguistics, 2015: 1343-1353.

[15] Stojanovski D, Strezoski G, Madjarov G, *et al. Twitter Sentiment Analysis Using Deep Convolutional Neural Network* [M]. Berlin Heidelberg: Springer-Verlag, 2015.

[16] Severyn A, Moschitti A. UNITN : Training deep convolutional neural network for twitter sentiment classification[J]. *Semeval*, 2014, **54**(2): 161-176.

[17] Santos d C N, Gatti M. Deep convolutional neural networks for sentiment analysis of short texts[C]// *Proceedings of COLING the* 25*th International Conference on Computational Linguistics*: *Technical Papers*. Dublin: Association for Computational Linguistics, 2014: 69-78.

[18] Chang J C, Lin C. Recurrent-Neural-Network for Language Detection on Twitter Code-Switching Corpus[EB/OL]. [2018-03-14]. *https://arxiv.org/pdf/*1412.4314.*pdf*.

[19] Joshi A K. Processing of sentences with intra-sentential code-switching[C] // *Proceedings of the* 9*th Conference on Computational Linguistics*. Prague: Association for Computational Linguistics, 1982: 145-150.

[20] Samih Y, Maharjan S, Attia M, *et al*. Multilingual code-switching identification via LSTM recurrent neural networks [C]// *Proceedings of the Second Workshop on Computational Approaches to Code Switching*. Austin: Association for Computational Linguistics, 2016: 50-59.

[21] Collobert R, Weston J, Karlen M, *et al*. Natural language processing (Almost) from scratch [J]. *J Mach Learn Res*, 2011, **12**: 2493-2537.

[22] Kim Y, Jernite Y, David S, *et al*. Character-aware neural language models [C] // *Thirtieth AAAI Conf Artif Intell*. Phoenix: AAAI Press, 2016: 2741-2749.

[23] Hinton G E, Srivastava N, Krizhevsky, *et al*. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors [EB/OL]. [2018-03-14].*https://arxiv.org/pdf/*1207.0580. *pdf.*

[24] James G, Witten D, Hastie T, *et al. An Introduction to Statistical Learning*: *With Applications in R* [M]. Berlin Heidelberg: Springer-Verlag, 2013.

[25] Brownlee J. What is the Difference Between Test and Validation Datasets?[EB/OL]. [2018-04-06]. *https*://*machine learningmastery.com/difference-test-validation-datasets*/.

[26] MacLachlan Geoffrey J, Do K-A, Ambroise K. *Analyzing Microarray Gene Expression Data*[M]. Hoboken:Willey, 2013.

[27] Al-Rfou R, Alain G, Almahairi A, *et al.* Theano: A Python Framework for Fast Computation of Mathematical Expressions [EB/OL].[2018-03-14]. *https://arxiv.org/pdf/*1605. 02688.*pdf.*

[28] Pedregosa F, Varoquaux G, Gramfort A, *et al.* Scikit-learn: Machine learning in Python[J]. *J Mach Learn Res*, 2012, **12**(10): 2825-2830.

□